installing other device drivers, hardware or software, are contemplated by the inventors to be within the scope of the invention.

[0039] Referring next to **FIG. 4**, an exemplary flow chart illustrates the operation of the driver software **304** components. At step **402**, the target software **110** is identified from a command received from a user. A list of one or more tasks associated with execution of the command is generated at step **404**. Generating a list of one or more tasks associated with the execution of a command occurs dynamically, or on-the-fly, in that the tasks are not hard-coded into the driver components. The driver components dynamically generate the list of tasks to be performed and store the list in a queue. At step **406**, the list of one or more tasks is stored in a queue accessible by the computer **102**. The list of one or more tasks stored in the queue are modified at step **408** such that the tasks are directed to operate on the target computer-readable media **108**. That is, the list is modified such that an operating system of the computer **102** such as operating system **302** will execute the listed tasks to manipulate or otherwise operate on the target software **110**. The driver components edit the list of tasks stored in the queue to point to the offline image loaded in the online system. For example, the driver component may edit a string in the list of tasks from "D:\i386\System32" to "C:\opsysdir\System32." At step **410**, the queue is committed to perform each of the modified tasks. Alternatively, the computer **102** may generate, store, and modify the tasks while the target software **110** is online and then commit the queue to perform the tasks when the target software **110** is offline.

[0040] Referring now to **FIG. 5, a** block diagram illustrates an exemplary computer-readable medium **502** storing a queue **504**. In this instance, the queue **504** is a data structure that can be used by the computer **102** to manipulate the target software **110** stored on the target computer-readable media **108**. The queue **504** has a queue field **506** that stores a list of one or more tasks **508** such as task #1 through task #N that are associated with execution of a command received from a user.

[0041] In one embodiment, the computer **102** modifies the list of tasks **508** stored in the queue field **506** to direct the tasks **508** to operate on the target computer-readable media **108**. Further, the computer **102** commits the queue to perform each of the modified tasks **508** stored in the queue field.

[0042] Referring next to **FIG. 6, a** block diagram illustrates the exemplary elements of software for offline manipulation of mass storage device drivers. As described above, the computer **102** is connected to the target software **110** via network **106**. In this example, the computer **102** includes user interface **602**, a system preparation file **604**, and a software tool **606**. The target software **110** includes a device database **608**. The device database **608** stores mass storage device drivers. While the computer **102** is shown connected to the target software **110** via network **106**, such a connection is optional as described above.

[0043] The user interface **602** is, for example, a graphical user interface (GUI) that allows the user to create the system preparation file **604**. In this instance, the system preparation file **604** includes a list of drivers **610** to be added to the target software **110** in response to the input received from the user, identification data **612** for each of the drivers, and configuration information **614** for each of the drivers. For example, the list of drivers **610** may include a text-based description of each driver. In addition, the identification data **612** may include a unique identifier corresponding to each driver. Further, the configuration information **614** may specify parameters for use when installing each driver.

[0044] The software tool **606** executes executable instructions on the computer **102** in response to input received by the user interface **602**. The software tool **606** includes a command-line interface **616** for parsing command-line options received via the user interface **602**. In one embodiment, the command-line options identify the system preparation file **604** (e.g., via a path), the target computer-readable medium **108** (e.g., via a path), and/or the target software **110**. The execution of the executable instructions by software tool **606** and the command-line interface **616** allows manipulation of the target software **110**. By way of example, and not limitation, the manipulation of the target software **110** includes: installing the list of drivers **610** to the target software **110**, enabling the drivers by adding the identification data **612** from the system preparation file **604** to the device database **608**, configuring the drivers according to the configuration information **614**, and updating one or more system settings for the server **104** as needed by the installed drivers.

[0045] In an alternative embodiment, the invention software is operable in an offline fashion with images or target software **110** that have not been fully applied, installed, or otherwise integrated. For example, the invention software can modify application programs that have been pre-installed or staged. Similarly, a software patch can be applied to an offline image prior to the first out of the box experience by the user. In a further example, an original equipment manufacturer (OEM) can add a driver that is necessary for booting from an offline image that includes an operating system. The offline image can then boot after the OEM adds the necessary driver. In another example, an OEM can add a service pack or other patch to numerous offline images that have not yet been applied or integrated.

[0046] **FIG. 7** shows one example of a general purpose computing device in the form of a computer **130**. In one embodiment of the invention, a computer such as the computer **130** is suitable for use in the other figures illustrated and described herein. Computer **130** has one or more processors or processing units **132** and a system memory **134**. In the illustrated embodiment, a system bus **136** couples various system components including the system memory **134** to the processors **132**. The bus **136** represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0047] The computer **130** typically has at least some form of computer-readable media. Computer-readable media, which include both volatile and nonvolatile media, removable and non-removable media, may be any available